Network Solutions.

Optimization as an Organizational Discipline

Dominic Delmolino
Network Solutions

Revised 15 February 2006

1 Introduction

An elusive goal since the end of the era when computing resources were scarce is the quest to tie computing resource consumption to actual monetary values. While it's been a Quixote-like quest of mine to re-introduce the notion of "chargeback" within an Oracle-based system, I've settled for implementing a process of tying performance optimization to business metrics and actual revenue growth as well as cost savings and containment. Based on my experiences as performance optimization consultant with Oracle Corporation from 1990 thru 2000, I decided to attempt to apply certain methodologies to a reference system over a sustained period of time. Since I joined Network Solutions as the director of database engineering, I've had the opportunity to implement these ideas. This paper describes my experiences in applying performance measurements and feedback mechanisms to Network Solutions' production database environment from mid-2002 thru early-2006.

2 Context

Before I begin discussing methods and results, I'd like to provide some context by way of defining what I mean by the terms *organization* and *optimization*. I'll also provide some background on the database systems and architecture under consideration.

2.1 Organization

Throughout this paper when I refer to the *organization*, I'm referring to a group of people who have both authority over and responsibility for the totality of Oracle resource consumption in a given production system. This totality includes architecture layout details like table and index structures, physical

placement instructions and instance configuration parameters. It also includes traditional resource consumption items like SQL and PLSQL[1] code. I understand that in some environments authority (the ability to make decisions and have them implemented) and responsibility (being held accountable for the results of decisions) are not contained within the same organization. I will attempt to address this concern in further detail below.

2.2 Optimization

When I refer to *optimization,* I will be talking a lot about efficiency. In my environment, efficiency refers to the amount of system resources consumed (disk space, disk access, CPU time, physical memory and network bandwidth) per *business* unit of work. Optimization is the process of making business units of work as efficient as possible – the goal being to decrease the amount of system resource required to perform larger and larger amounts of business work units (in both size and quantity). This process requires the ability to measure business units of work, system resource consumption and the means to correlate them.

2.3 Systems and Architecture

Network Solutions' main web site provides customers with product purchase and management functions for web-based solutions, primarily domains, but most recently adding more enhanced email and web site hosting. In mid-2002, Network Solutions' web site was supported by 8 full-size Oracle databases (3 small order-processing databases, 1 large order history database, 1 large customer and product database, 1 billing database, 1 database to support one product and 1 database providing customer support data). Database access was routed through a Java middle-tier and most access was performed through stored procedures. Business logic resided in both the front-end (web) and middle (Java) tiers.

3 Defining the Organization

3.1 The Traditional Organization

Companies that rely on internally developed application software for unique competitive advantage typically segment their technical organizations into Operations Support (other names include IT, Production Support, and Operations Management) and Application Development (other names include Software Development, Engineering or MIS). Sometimes an independent Quality Assurance and Testing group exists, but more often than not, such a function is contained within the Application Development group.

Databases, which have a high degree of operational support requirements in addition to their application capabilities, are usually given over to the Operations Support group. Database application development (schema design, SQL and PLSQL coding) is reserved for the Application Development group and is

[1] I know it's PL/SQL, but I'd like to reserve the / for other ideas within the document text. So, from here on out, read PL/SQL for PLSQL if it makes you feel better.

usually performed by application developers for whom database skills form part of their secondary skillsets[2].

The people responsible for the databases are referred to as database *administrators* (DBAs) (emphasis intentional) and usually have relatively limited authority over decisions concerning database resource usage. Responsibilities generally include:

1. Availability
2. Space Management
3. Performance Management, as (narrowly) defined by:
    a. Systemic Tuning (via configuration parameters)
    b. Performance Monitoring

While availability can properly be relegated to the operations staff with full authority, the other areas of responsibility usually do not coincide with a commensurate level of authority.

It's this disconnect of having full responsibility for space and performance management without a significant, authoritative voice in the development of space and performance decisions which usually cause problems in database performance. Most DBAs recognize that application developer decisions regarding database schema design and SQL / PLSQL coding have significant impact on database performance. Unfortunately, DBAs are limited in their ability to affect such decisions.

I believe there are several traditional reasons and/or myths given by companies for these limitations:

1. Decisions about SQL / PLSQL coding belong in the Application Development group.
2. Decisions about schema design and layout belong in the Application Development group.
3. Space management is an Operations group problem.
4. Performance management can always be addressed via System Tuning of system and configuration parameters.
5. Database administrators are responsible for *administering* a system, not *designing or developing* it.
6. Feedback mechanisms between Operations and Application Development are stunted within the company due to:
    a. Physical separation of the organizations
    b. Differences of opinion and/or mission goals regarding cost-containment vs. revenue growth.
    c. Generational differences between the organizations (e.g., Operations folks refer to DASD, Application Development talks about SANs).
    d. Stereotypical connotations around the title of Administrator (i.e., equating Database Administrators with System Administrators).

These limitations, in combination with the responsibility / authority disconnect[3], are the main reasons I see for some of the rancor and emotion around database performance issues and their lack of resolution within many companies.

---

[2] Their primary skillsets usually being interactive front-end languages or service-layer languages.

3.2 An Optimal Organization

If I were designing an organization from scratch, I would create a database team within the Application Development organization with both the full authority and responsibility to make all database resource decisions (schema design and SQL / PLSQL coding). Simply clearing up the *accountability* for the results of these decisions will make handling of performance issues that much easier.

I'd call the members of the group Database Engineers (DBE's) in order to overcome any stigma associated with the *administrator* term.

I'd make sure the DBE team had a good working relationship based on mutual respect and trust with the DBA team in Operations. I can't stress this enough – in the end, the Operations team is essentially the customer of the Application Development team. They are the ones who have to live with and manage the consequences of the decisions made by the DBE team – especially when problems occur in the middle of the night, on holidays, when everyone else is out partying.

All database access code would be controlled by this team. All SQL. All PLSQL. All access to the databases would be via stored procedures – religious disagreements over the location of business logic would be avoided as much as possible (we'll discuss strategies for this later).

We'd strive for operational efficiency by removing obstacles and any activities which could be automated. We'd develop basic standards so that code could be easily shared, critiqued and improved by team members regardless of who initially wrote it.

I'll go into details later on in this paper, but for now, let's talk about some objections and potential ways to overcome them.

3.3 Overcoming Objections

Usually, I find three main groups who object to the proposed organization:

1. **Management**. Who have to pay for the new team and deal with the political fallout of taking functions away from other teams.
2. **Application Developers**. Who fear a loss of control if they have to rely on another team to get the database work done – especially if they feel that they don't need another "software layer". They also may resent the implication that they aren't competent at handling simple database access code[4].

---

[3] I'm pretty passionate myself about how responsibility and authority are divided up within organizations and upon individuals. Anyone who's given a job where they have more responsibility than authority will (in my opinion) quickly become emotionally non-productive. An overdose of authority without responsibility causes some well-known problems too.

[4] Part of this comes from what I believe to be some of the original goals of SQL – "to make it possible for non-programmers to interact with databases". Which has led to the belief that SQL isn't a "real" programming language, and, as such, can be done "part-time" by "real" programmers. The above quote comes from a fascinating edited transcript of discussions at a 1995 reunion of folks who

3. **Operations**. Who want a chance to have a say in what decisions will be made and feel that they won't be able to if a new team in Applications Development is created.

Management can usually be appealed to on the grounds of *accountability*. It's usually easy to point out how the current organizational structures aren't working out that well – performance problems aren't resolved quickly, there's a lot of inter-organizational finger-pointing and a lot of expensive consultants called in. By creating a central, responsible team with the authority to make database decisions, management can feel better about whose task it is to deal with database performance issues. Of course, one must assemble a competent team, but that should be a straightforward task – form a team, and ruthlessly hold them accountable.

Application Developers can usually be appealed to on the grounds of *relievement* summed up in the phrase "We'll take care of that for you". Generally it's easy to point out currently existing inefficient SQL and PLSQL code and help the developers understand that it's just not their forte. "Just tell us want you want from the database and we'll take care of it." – is a reasonably powerful message. You can appeal "vanity" too by telling them they can focus on more "important" stuff too. I've also found that by eliminating embedded database code from applications, you can avoid the need for recompilations when performance changes are necessary.

Finally, you need to commit to involving Operations in the process. Understand that in the end, they can be a voice complimenting your work or constantly complaining about it. By making them feel like partners in the development of the database, you can make sure they are on-board with having a dedicated group listening to their concerns.

A word about business logic

If you're able to overcome all of the objections listed above, consider yourself well on the way to success. This is not the time to get into a religious argument over where business logic *should* reside. You may need to "trade-off" on your desire to put all business logic in the database against the ability to control all SQL and PLSQL. And honestly, once you have the SQL and PLSQL, you can slowly transition business logic into the database. This is another area where you can use the "We'll take care of it for you" mentality – especially when developers insist on a pure, public, "data" API which gets abused by applications[5]

---

worked on System R and its derivates (http://www.mcjones.org/System_R/SQL_Reunion_95/sqlr95-System.html)

[5] I don't have room for it in this paper, but there can be substantial performance differences between a "data" API and what I call a "functional" API. Applications have a habit of preferring a "data" API, in which they treat the database like a flat file and loop through records one at a time. Database engineers cringe at this approach to set processing – especially when the supposed "performance" improvements include adding more threads or buying more powerful hardware.

4 Continual Optimization

4.1 Inspiration

"Most customers have 5x more system capacity for their database servers than they actually need." (Tom Kyte, *lunch* [2001])

"Less code = less bugs" (John Beresniewicz, *quote on whiteboard at Savant* [2001])

"The number you're looking for is 5 LIOs per row per join" (Cary Millsap, *email conversation* [2003])

4.2 Organizational Application

Once you've gotten your optimal organization, what can you do to ensure that it stays focused on developing efficient, optimal SQL and PLSQL code?

Here are the main organizational processes we've applied in our organization:

1. Standard IDE tools
2. Rigorous, streamlined source-code control
3. Specific tuning processes and goals
4. Requirements negotiation
5. Mental snacks

I'll go into detail on each one.

Standard IDE Tools

When I joined Network Solutions, everyone on the team was using a different set of "tools" to interact with the database. Some folks used *vi* and Unix-based *sqlplus*. Others used Toad from Quest Software. One person used SQL Navigator from Quest Software. From my prior jobs I had experience with PLSQL Developer from All Around Software.

I wanted everyone to use the same tool in order to foster the ability for people to collaborate more easily – when looking over a colleague's shoulder it's easier when you're familiar with the tool being used.

We chose Toad due to the following requirements I had:

1. Includes DBA functions (space management, tablespace management, redo log management) in addition to basic schema browsing (and object creation) as well as a PLSQL coding environment.
2. Includes useful source code and schema comparison tools including difference reports, single-object diff ability and schema synchronization scripting.
3. Includes basic source-code control over in-database object manipulation (implements a basic check-out / check-in method on all PLSQL objects).

TOra was an attractive alternative before it was bought and retired by Quest. Going forward, I have high hopes for Oracle Project Raptor.

Rigorous, Streamlined Source Code Control

Source code control is paramount in our environment – we need to quickly identify differences between our Development, Test, QA and Production environments and be able to talk about why such differences exist and when they were introduced.

At Network Solutions, nothing goes into Production without first being installed and tested in Test and QA environments. So the ability to deliver database scripts which can be moved through multiple environments is important.

Prior to my arrival, all database "upgrade" scripts were hand-coded and simply checked into our file-based source code control system. The hand-coded extracts of code from Development were non-standard and often error-prone as each Database Engineer did them individually. Also, each script would contain changes to multiple objects – making it difficult to isolate which script affected which object. Hand-coding these scripts was also time consuming.

I instituted a policy that all database scripts would be extracted directly from the development database using code generation scripts run from *sqlplus*. Each script would be for one and only one database object at a time. Since we were running Oracle 8i, I wrote up *sqlplus* scripts which extracted DDL commands based on the data dictionary[6]. When we upgraded to Oracle 9i, we converted most of the scripts to use DBMS_METADATA[7]. Script names contained information about what was being done. Here are some examples:

mk-tbl-edb-prod_cd.sql (creates a table in the edb schema called prod_cd).
ch-tbl-edb-prod_cd-33172.sql (alters a table in relation to ticket# 33172).
rm-tbl-edb-prod_cd.sql (drops the prod_cd table).

By using some basic Unix-like prefixes, we were able to indicate to the production DBAs what was being done in each script.

These code generation scripts are included in Appendix A.

In addition to the code generation scripts, we fully audit all DDL commands against our databases. By turning on DDL auditing at the database level, I can see every structural change introduced into the system, who did it and when[8].

Specific Tuning Processes and Goals

We don't use any fancy tools here. Just simple performance metrics and basic Oracle utilities. Since, for the most part, our system is an OLTP system, queries need to pass a simple timing test – it either comes back quickly or it doesn't. If it takes more than 10 seconds, it's too slow. ☺

---

[6] A further benefit of this approach was the ability to inject code during extracts. We inject SCCS-like tags and comments.
[7] DBMS_METADATA rocks!
[8] It amazing how under-utilized Oracle DDL auditing is. People try to get all clever with schema triggers when the basic capability is built right into the database. Some people worry about the performance impact – what kind of system has so many DDL commands? ☺

Beyond that, we use a simple metric – *logical i/o operations per row retrieved per row source (usually table)*. Our ultimate goal is usually 5 LIOs per row per join (thanks, Cary Millsap!). DBE's are required to tune their queries with this metric in mind. We usually use simple *set autotrace traceonly* commands in *sqlplus* to accomplish this.

I regularly scan the contents of *v$sqlarea* using a simple report (included in Appendix B) to see how well we are doing in meeting this goal.

Requirements Negotiation

Since this group now handles all SQL and PLSQL, we regularly get requests from application developers and clients for new queries. I've worked to instill an ethic of asking about the purpose for each query instead of blindly working to fulfill each request as soon as possible.

The reason for this is that the application developers typically want a "data" API in which we simply return result sets and they use business logic in their tier to decide on a course of action. However, there are usually a couple of problems we encounter when we don't question that approach:

1. The application developer wants to treat the database as a flat file of records, in which they loop over result sets one record at a time.
2. It's often more efficient for us to perform the function based on what we see / find in the data inside a stored procedure.

Single record at a time processing by the application tier represents a "hidden" performance problem for us. Individual queries may be super-fast (and adhere to our tuning metrics), but we'll problems as the applications start invoking them at high rates. And, unfortunately, some of our application developers react to perceived performance problems by simply "multi-threading" the process in order to make it go faster. In our system, the classic example of this is our renewal notice generation process, in which we generate emails for clients to let them know that their products are expiring. Our original requests for database code came in like this:

1. "Give me a list of all customers who have products expiring within the next 2 weeks"
2. (A few days later) "For a given customer, give me all of their products which are expiring within the next 2 weeks".

The DBEs implementing these requests did not originally ask the purpose behind each one, how they might be related, and at what frequency they'd be used. Implemented independently, each request was written efficiently and performed well isolation. And actually, we thought that request #2 (the products for a specific customer) was going to be used in the customer service application when investigating a particular customer. Unfortunately, the daily batch job generating renewal emails decided to call query #1 and then immediately loop over the results calling query #2 for each customer individually. They'd then further work by calling an update procedure to indicate that they had sent renewal notices for each product. Time to complete in production: 8 hours each night.

We re-wrote this to have the database do the entire job in one call, both returning a result set (ordered by customer) and setting the renewal notice flags. New time to complete in production: 40 minutes.

The key here was understanding the requirements and working together with the application developer to come up with an optimal solution from both ends.

Mental Snacks

Finally, we make sure we're never too busy to take time out for mental calisthenics. Whenever we encounter either a common coding error or an interesting coding approach, we take the time to educate the whole team.

These mental snacks take many forms:

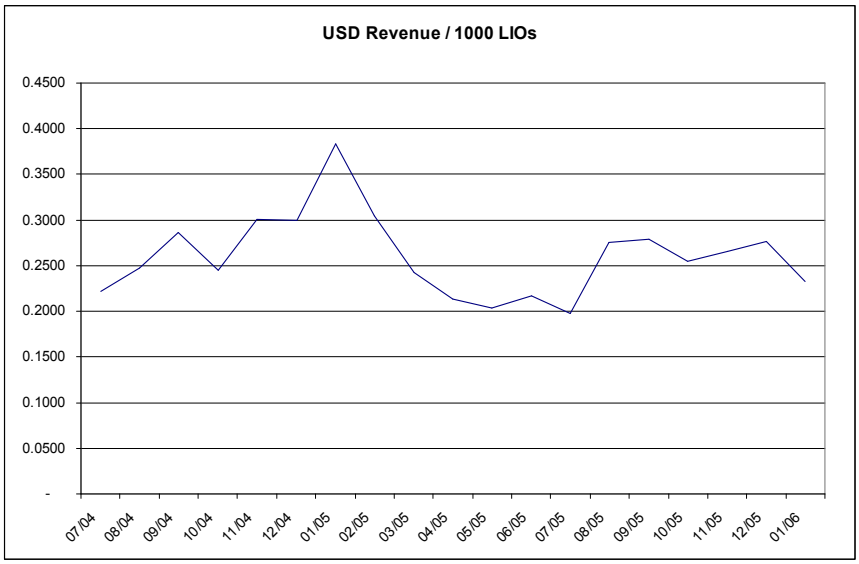
1. An impromptu meeting to review a problem and solution.
2. An email quiz question intended to highlight an issue
3. A micro-project presented as a contest

My staff particularly like the micro-project contests – usually a task that should take a DBE no more than 4-6 hours. I usually award the "winner" an Amazon gift certificate – for which I suggest relevant Oracle books.

The most important take-away from these mental snacks is the group-wide education. I like to make sure that everyone can explain the goal of the exercise and why certain solutions are better than others. I also think the staff like the "change of pace" provided by this mini-break.

5 Results

At the beginning of this paper I mentioned how I've been trying to tie resource consumption to actual monetary values. I'd like to discuss other results as well, but here are some basic figures from the beginning of July 2004 through the end of January 2006.



From July 2004 through January 2005, you can see the results of our tuning efforts where we were able to increase $/LIO by 70%. In January 2005, Network Solutions starting selling hosting packages which included a website and (most importantly) between 25 and 100 email boxes per package (most of

which are unused by the customer). We made an implementation comprise by implementing hosting email boxes the same way we had implement so-called ala carte email boxes in the past – as full, instantiated products in their own right. As a result, our product count exploded, but did not remain in line with our revenue collection. This essentially wiped out all of our tuning efforts from 2004. ☹

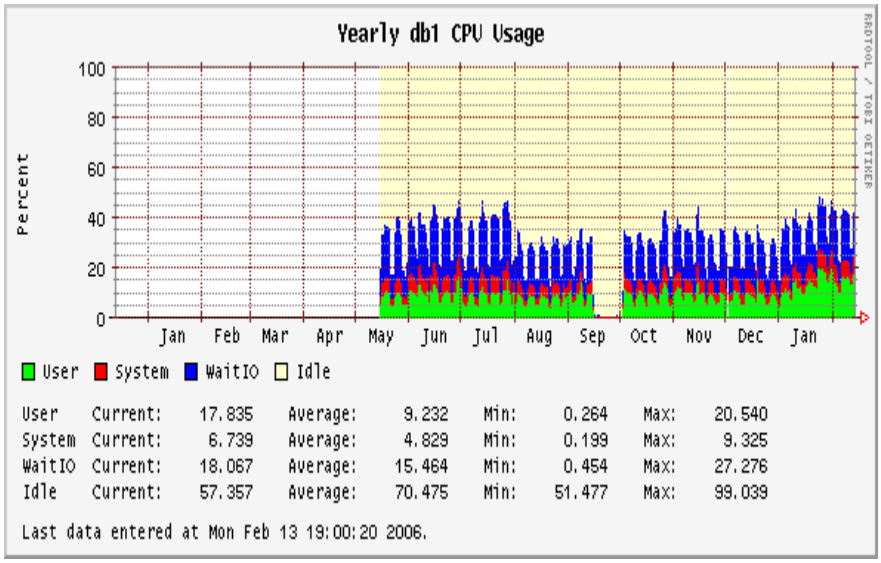
In July 2005, we deployed several tuning changes to some of the SQL around the hosting packages. Most notably, we dramatically improved the queries related to allowing customers to configure new email boxes. This increased our $/LIO by 40%.

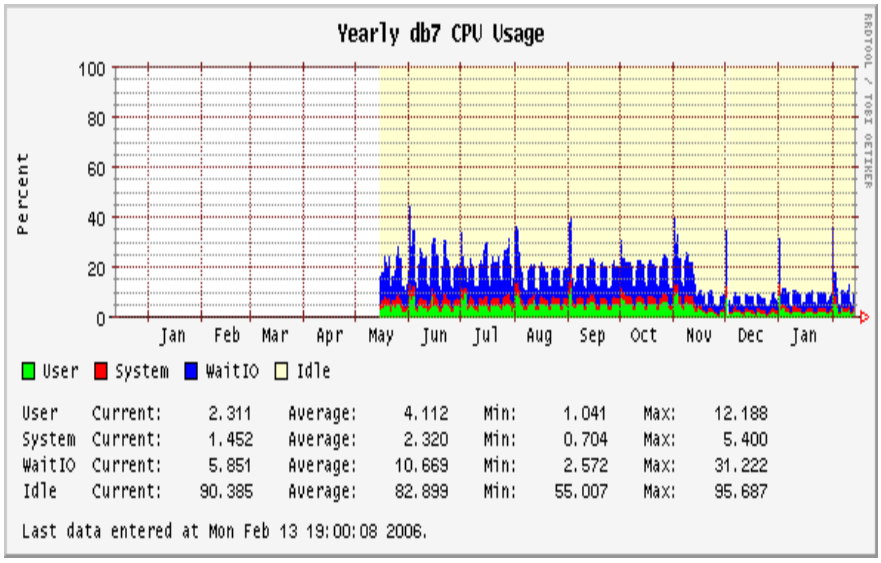
In May 2006, we plan on no longer instantiating hosting email boxes at the time of purchase, but rather creating them on demand as customers configure them. This should have a dramatic impact.

Another area of improvement we've been working on is decreasing the number of statement executions done by the server (i.e., we'd like to increase the size of the units of work and reduce the back-and-forth chatter between the database and the application servers).

Most of the improvement here came in the November 2005 release where we were able to consolidate many stored procedures and queries. The apparent "improvement" in January 2005 was due to the larger payload associated with a hosting package – retrieving a single hosting package (1 exec) usually returned a lot of "products' (associated email boxes).

Another way to look at our progress is to measure CPU utilization on the main database servers. Here is yearly data from our main Customer/Product (db1, also known as EDB) and Billing (db7, also known as SVCB) database servers:

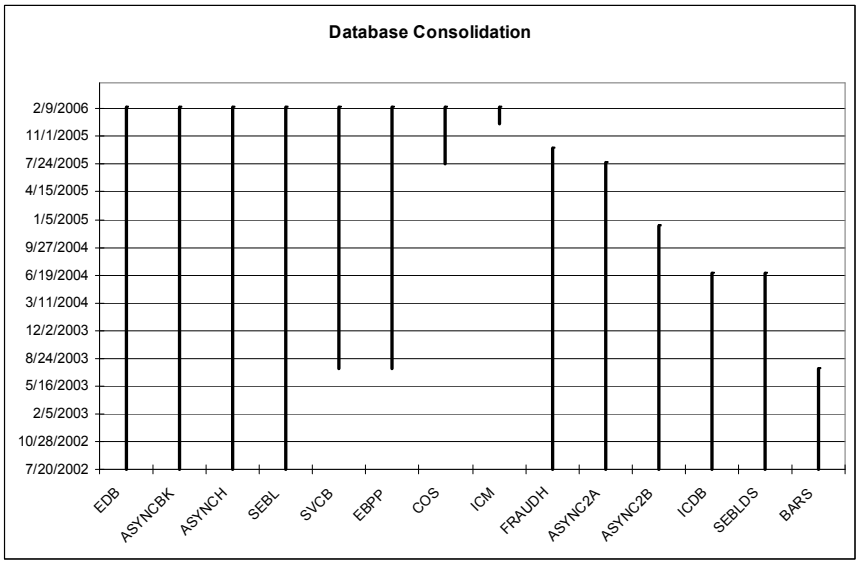




Here, you can see the effects of our end of July 2005 release (on db1) (where we tuned the email configuration query), and our November 2005 release (on db7) (in which we consolidated data into db1 from db7).

Prior to the November 2005 release, we used data from our Billing database to authoritatively indicate product expiration dates – this necessitated a distributed join between db1 and db7 in order to present the full picture of each product. In the November 2005 release, we consolidated that data into db1 in order to eliminate the dependency on db7 for customer product queries.

Another area of optimization we've been working on is database consolidation as shown below:



Since July 2002, we've eliminated 6 databases and added only 4 (3 of which are COTS Vendor-based, which I like to keep separate for certification and version reasons). The COS and EBPP databases are targeted for elimination next.

Our source code control procedures have been a universal hit, with our delivery packages looking something like this:

```
v009005/ddl/ch-tbl-edb-alias-35063.sql
v009005/ddl/ch-tbl-edb-email_box-35063.sql
v009005/ddl/ch-tbl-edb-fs_service_type_group-34755.sql
v009005/ddl/ch-tbl-fmt-fmt_product-35114.sql
v009005/ddl/mk-idx-edb-domain_prereg_req_ik_reqstat.sql
v009005/ddl/mk-idx-edb-domain_prereg_req_uk_domaccrt.sql
v009005/ddl/mk-idx-edb-email_box_uk_dnsmbox.sql
v009005/ddl/mk-idx-edb-fs_service_type_group_pk.sql
v009005/ddl/mk-idx-fmt-fmt_order_ix_merchant.sql
v009005/ddl/mk-idx-fmt-fmt_order_ix_processing_state.sql
v009005/ddl/mk-idx-fmt-fmt_rejected_ip_ix_inet.sql
v009005/ddl/mk-tbl-fmt-fmt_test_dates.sql
v009005/ddl/rm-idx-fmt-fmt_order_ix_merchant.sql
v009005/plsql/mk-spb-edb-cc_pkg_v9_5.sql
v009005/plsql/mk-spb-edb-edb_rasii_v9_5.sql
v009005/plsql/mk-spb-edb-optimizedfgpm_pkg_v9_5.sql
v009005/plsql/mk-spb-edb-prod_inst_wholesale_trg_pkg.sql
v009005/plsql/mk-spb-edb-sm_pkg_v9_5.sql
v009005/plsql/mk-spb-fmt-fmt_investigate_order_pkg_v95.sql
v009005/plsql/mk-spb-fmt-fmt_netsol_import_pkg_v95.sql
v009005/plsql/mk-spb-fmt-fmt_run_rules_pkg_v95.sql
v009005/plsql/mk-spb-fmt-fmt_test_import_pkg_v95.sql
v009005/plsql/mk-spb-pricing-pricing_pkg_v9_5.sql
v009005/plsql/mk-sps-edb-cc_pkg_v9_5.sql
v009005/plsql/mk-sps-edb-edb_rasii_v9_5.sql
v009005/plsql/mk-sps-edb-optimizedfgpm_pkg_v9_5.sql
v009005/plsql/mk-sps-edb-sm_pkg_v9_5.sql
v009005/plsql/mk-sps-fmt-fmt_investigate_order_pkg_v95.sql
v009005/plsql/mk-sps-fmt-fmt_netsol_import_pkg_v95.sql
v009005/plsql/mk-sps-fmt-fmt_run_rules_pkg_v95.sql
v009005/plsql/mk-sps-fmt-fmt_test_import_pkg_v95.sql
v009005/plsql/mk-sps-pricing-pricing_pkg_v9_5.sql
v009005/plsql/mk-trg-edb-alias_briu_lowercase.sql
v009005/plsql/mk-trg-edb-email_box_briu_lowercase.sql
v009005/dml/ins-edb-fs_product_type.sql
v009005/dml/ins-edb-fs_service_type_group.sql
v009005/dml/ins-edb-prod_cd.sql
```



```
v009005/dml/ins-edb-prod_composition.sql
v009005/dml/ins-pricing-nsi_prod_display.sql
v009005/dml/ins-pricing-nsi_prod_display_map.sql
v009005/dml/ins-pricing-service_group_members.sql
v009005/dml/ins-pricing-vrsn_prod_composition.sql
v009005/dml/ins-tbl-edb-amp_reward-35134.sql
v009005/dml/ins-tbl-edb-amp_reward_channel_xref-35134.sql
v009005/dml/ins-tbl-edb-country_list-35273.sql
v009005/dml/upd-edb-prod_composition.sql
v009005/dml/upd-pricing-nsi_prod_display_map.sql
v009005/dml/upd-tbl-edb-email_box-35063.sql
v009005/dml/upd-tbl-edb-prod_inst-35063.sql
v009005/install/install-edb-v95-out-100.sql
v009005/install/install-edb-v95-out-110.sql
v009005/install/install-edb-v95-out-120.sql
v009005/install/install-edb-v95-pre-100.sql
v009005/install/install-fmt-v95-pre-100.sql
v009005/install/install-pricing-v95-out-200.sql
```

From this kind of file list, our production DBAs and QA personnel can relatively easily see what database objects and object types are being deployed.

In addition to this kind of installation packaging (made possible by our code generation scripts), we have our init.ora file under full source code control for parameter history tracking (yes, we use pfiles for this instead of spfiles) (See Appendix C for example).

Finally, I'm fairly proud of the performance efficiency improvements my staff have made in the SQL that runs against our core Customer/Product system:

Hash Value	LIO / Exec	PIO / Exec	CPU / Exec	Joins	LIO / Row / Join	Execs	Rows	Gets	PIOs	CPU Time	Elapsed Time	Elap / CPU
460974741	7	0	7	3	250	5391109	37811	37806269	105107	37438090	38694500	1.03
2890208721	21	0	12	8	2	1762808	1762802	36586656	25833	21835820	24747555	1.13
2425919778	16	0	4	5	3	5084473	5084068	81434909	19147	21606450	21624667	1
1831825336	15	0	1	6	1	14829392	35251411	216930007	2384265	13723860	33192358	2.41
2747937127	23	1	11	10	2	1183896	1183861	27607330	870344	12982940	23072223	1.77
2064822940	10	0	1	3	2	14832194	20727091	144235173	576378	10443230	14837616	1.42
3911951598	25	1	10	7	1	1003376	2403661	25037613	761581	10085040	18072733	1.79
1954278109	51	1	5	2	17	2013152	2013141	103227016	1964262	10076700	35959130	3.56
2032056795	7	0	2	2	2	5791601	5791592	37892672	826036	8858910	17126390	1.93
2033996523	13	1	11	7	1	761207	1101107	10267739	576203	8304350	14011604	1.68
4248548543	355	20	10	4	24	669644	1995150	238029442	13145808	6841950	103095306	15.06
203638781	130	0	9	10	0	744766	19974724	96925170	356669	6826770	9680997	1.41
3306219282	7	0	0	2	0	13822454	83699995	99171097	3031348	6797870	22275596	3.27
1990360629	10	0	1	2	2	4821007	7742416	46053335	743957	6281430	15571873	2.47
187817516	21	0	1	2	7	5519462	5509126	115163611	1533930	5834650	20731005	3.55
2284357310	13	0	0	4	1	15760322	62190137	200779313	3542585	5719300	40701247	7.11
1421068162	30	1	13	0	30	400106	400106	12168336	515798	5318660	13010530	2.44
1691573251	96	8	22	35	1	233920	455759	22402834	1862581	5205970	25038350	4.8
2390646151	30	1	12	0	30	400106	400106	12166592	515783	4923710	12585112	2.55
908878421	166	10	39	4	29	118409	134122	19627307	1217738	4675300	5739427	1.22
21407430	8	0	0	3	0	12091644	78919712	98135055	3270602	4666600	38752750	8.3
1981243314	250	30	14	6	36	337938	337938	84348244	10241965	4583660	123589118	26.96
2136071715	404	6	19	7	1	242088	9637463	97772403	1506699	4511590	20400976	4.52
3927009719	128	54	10	2	38	416995	470069	53310690	22373342	4158780	52280886	12.57

3098044555	182	11	14	36	2	286482	590419	52102470	3191292	4111180	29212877	7.1
326359704	9	0	0	9	1	15834133	15833911	141561748	178	4049340	4066727	1
1510873260	37	1	25	17	2	140081	140081	5149876	130354	3525160	4404895	1.24
1815913401	37	0	6	8	2	532073	1078726	19662547	4189	3344180	3327154	0.99
1038497600	62	8	10	10	0	337937	4011882	20980212	2638825	3289630	28157972	8.55
2470152341	763	14	14	11	2	242072	7595938	184594316	3360570	3274080	39331532	12.01
2286960274	29	1	13	12	2	258888	258887	7574479	204286	3236440	6063521	1.87
3779016779	17	2	1	8	1	3897313	8113312	66930456	5977139	3113320	44885109	14.41
962389367	8	0	0	4	2	13822424	13821888	111841349	719506	3109990	10078855	3.24
2460565101	178	9	16	15	1	183116	3559170	32531556	1722777	2933240	18722232	6.38
4293856379	3	0	0	4	34	12091652	234236	39955817	4038056	2918460	39675720	13.59
337584979	157	1	12	11	0	242088	8272196	37979445	224200	2878440	5362553	1.86
4271637780	5	0	0	2	2	15979048	15978824	79927148	176	2869250	2747668	0.95
229119125	246	0	4	1	1	669616	161614487	164724042	352	2851310	2438551	0.85

While not perfect, many of the "top" statements are just about as efficient as possible, with LIOs / Row / Join between 1 and 10.

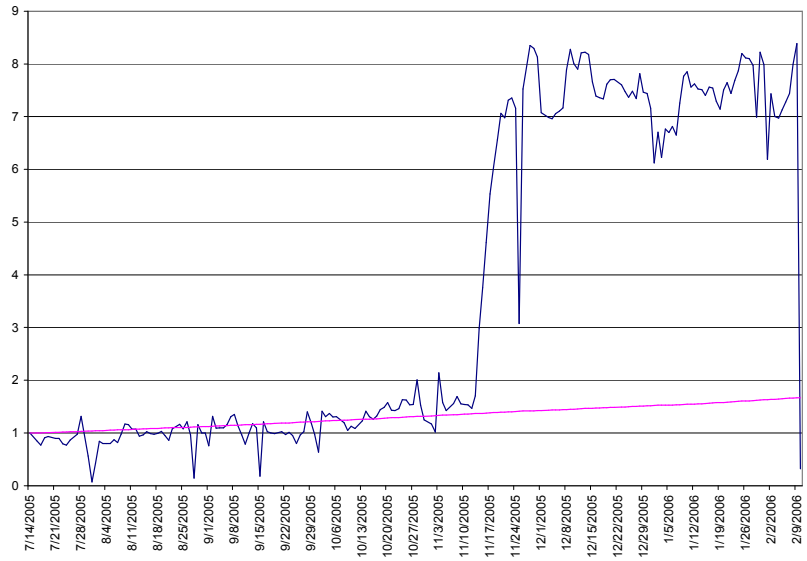
6 Challenges

Our process isn't perfect, there are still some challenges we're dealing with:

6.1 "Public" API Abuse

One thing I'm always asked for by our Java developers is a "simple" public API (insert / update / delete / select w/XML filtering capability – one set of procedures for every table). The thought being that with a public API, they can easily construct the business logic themselves without having to wait for a custom stored procedure. The biggest problem we have with that approach is the serious potential for abuse by developers accustomed to flat-file looping algorithms (and "speedups" by adding multi-threading). While this problem can also occur in APIs which are more business-function based, we see more abuse on the generic APIs than anything else in our system.

Recently, we noticed that the system seemed to be running hot over the past few months and were able to trace part of the problem to a generic statement which suddenly was being called 8x more than it normally was.



We've traced this to a "public" API (query_product), but so far are unable to determine what change in our November 2005 release caused this issue.

Unfortunately, we didn't discover this until recently (3 months after deployment) – partially because I focus too much on efficiency and not on call volume. And our only repository of call volume history is hourly PERFSTAT gathering. I'm looking forward to using AWR in 10g in order to catch this kind of this earlier.

6.2 The Long Tail

At this point most of our statements are pretty well-tuned, and yet we are still looking for tuning opportunities. Part of the problem is that top slowest queries are accounting for a smaller and smaller performance problem when considered against the total number of statements in our system. Unfortunately, it's difficult to analyze the very long tail of statements – our current approach is to reduce the number of distinct statements, but I'm not particularly happy with our progress to date.

7 Acknowledgements

I'd like to thank all of the people who've given me the opportunity to explore the implementation of optimization techniques in large-scale Oracle systems. Cary Millsap for mentoring me during my days at Oracle Corporation. Virag Saksena for crazy technical discussions about queuing theory as applied to waiting in line for inner tubes at Disney's Blizzard Beach. Brian Kush for keeping me focused on results. Don Andersen for a wonderful argument about chargeback. Jon Beresniewicz for keeping me focused on simplicity. Tom Kyte for reminiscing with me about the days when computing resources were expensive and perhaps still are. My colleagues and staff at Network Solutions who provide me with tons of support and encouragement and productive debates: Mike Cocozza, Mona Nahavandi, Samir Badalov and my manager, Pete Fox. And finally my beautiful wife and children – Wendy, Peter and Francesca – for their patience, devotion and ability to provide me with a true perspective on life.

8 About the Author

Dominic Delmolino is the Director of Database Engineering at Network Solutions. At Network Solutions, Mr. Delmolino oversees the design and development of the Oracle databases which support Network Solutions main customer-facing websites.

Mr. Delmolino served within Oracle Corporation for 10 years, where he participated in over a hundred Oracle Consulting projects and taught performance optimization and data replication topics to several hundred consultants, salespeople and Oracle customers. After a brief stint as CTO of Savant Corporation in 2000, he joined Network Solutions in 2002.

9 Revision History

17 February 2006: Created in preparation for the *Hotsos Symposium* 2006.

10 Appendix A

10.1 Mk-tbl

```
rem
rem     mk-tbl
rem

ttitle off
btitle off
set pause       off
set heading     off
set feedback    off
set verify      off
set echo        off
set termout     off
set recsep      off
set trimspool   on
set pagesize    0
set linesize    160
set long        16384
set longchunksize 16384

column  file_col        new_value       output_file     noprint
column  prog_line       format a70
column  const_line      format a70      newline
column  cmt_line        format a70      word_wrapped
column  one_line        format a1000    trunc
column  ord_remark      noprint
column  ddl_cmd         word_wrapped

rem
rem     Spool to output file
rem

select  lower('mk-tbl-' || '&&1' || '-' || '&&2' || '.sql')
file_col
from    dual;

spool   &output_file
```

```
select  'rem'                           const_line,
        'rem %TC-INFO%'                 const_line,
        'rem ' || '&&3'                 const_line,
        'rem'                           const_line,
        ''                              const_line,
        'set define off'                const_line,
        'prompt Creating table ' ||
        upper('&&1') || '.' ||
        upper('&&2')                    const_line,
        ''                              const_line
from    dual;

exec
dbms_metadata.set_transform_param(dbms_metadata.session_transform,'
SQLTERMINATOR',TRUE);

select
dbms_metadata.get_ddl('TABLE',upper('&&2'),upper('&&1'))
from    dual;

select  ' ' const_line,
        'create public synonym ' || upper('&&2') const_line,
        ' for ' || upper('&&1') || '.' || upper('&&2') || ';'
const_line
from    dual;

select
dbms_metadata.get_dependent_ddl('OBJECT_GRANT',upper('&&2'),upper('
&&1'))
from    dual
where   exists
(select 'x'
from    dba_tab_privs
where   owner = upper('&&1') and table_name = upper('&&2'));

select
dbms_metadata.get_dependent_ddl('COMMENT',upper('&&2'),upper('&&1')
)
from    dual
where   exists
(select 'x'
 from   dba_tab_comments
 where  owner = upper('&&1') and table_name = upper('&&2') and
table_type = 'TABLE' and comments is not null)
or      exists
(select 'x'
 from   dba_col_comments
 where  owner = upper('&&1') and table_name = upper('&&2') and
comments is not null);

select  ' ' const_line,
        'begin' const_line,
        'dbms_stats.gather_table_stats(' const_line,
        'ownname=>''' || upper('&&1') || ''',' const_line,
        'tabname=>''' || upper('&&2') || ''',' const_line,
        'cascade=>TRUE' const_line,
        ');' const_line,
        'end;' const_line,
        '/' const_line,
        ' ' const_line
from    dual;

spool off
```

10.2 Mk-sps

```
rem
rem     mk-sps
```



```
rem

ttitle off
btitle off
set pause       off
set heading     off
set feedback    off
set verify      off
set echo        off
set termout     off
set recsep      off
set trimspool   on
set pagesize    0
set linesize    79
set long        16536

column  file_col        new_value       output_file     noprint
column  prog_line       format a70
column  const_line      format a70      newline
column  cmt_line        format a70      word_wrapped
column  ord_remark      format a1                       noprint

rem
rem     Spool to output file
rem

select  lower('mk-sps-' || '&&1' || '-' || '&&2' || '.sql')
file_col
from    dual;

spool   &output_file

select  'rem'                           const_line,
        'rem %TC-INFO%'                 const_line,
        'rem ' || '&&3'                 const_line,
        'rem'                           const_line,
        ''                              const_line,
        'set define off'                const_line,
        'prompt Creating package ' ||
        upper('&&1') || '.' ||
        upper('&&2')                    const_line,
        ''                              const_line,
        'create or replace'             const_line
from    dual;

set linesize 4000

select  text, case when line >= 2 then line+1 else line end
ord_remark
from    dba_source
where   owner = upper('&&1')
and     name = upper('&&2')
and     type = 'PACKAGE'
union
select  '/* %TC-INFO% */', 2
from    dual
order by 2;

select  '/'                             const_line,
        ' ' const_line
from    dual;

select  'create public synonym ' || upper('&&2') const_line,
        ' for ' || upper('&&1') || '.' || upper('&&2') || ';'
const_line,
        ' ' const_line
from    dual;

select  'grant ' || privilege || ' on '         const_line,
        owner || '.' || table_name              const_line,
        'to ' || grantee || ';'                 const_line
```



```
from    dba_tab_privs
where   owner = upper('&&1')
and     table_name = upper('&&2');

spool off
```

10.3 Mk-spb

```
rem
rem     mk-spb
rem

ttitle off
btitle off
set pause       off
set heading     off
set feedback    off
set verify      off
set echo        off
set termout     off
set recsep      off
set trimspool   on
set pagesize    0
set linesize    79
set long        16536

column  file_col        new_value       output_file     noprint
column  prog_line       format a70
column  const_line      format a70      newline
column  cmt_line        format a70      word_wrapped
column  ord_remark      format a1                       noprint

rem
rem     Spool to output file
rem

select  lower('mk-spb-' || '&&1' || '-' || '&&2' || '.sql')
file_col
from    dual;

spool   &output_file

select  'rem'                           const_line,
        'rem %TC-INFO%'                 const_line,
        'rem ' || '&&3'                 const_line,
        'rem'                           const_line,
        ''                              const_line,
        'set define off'                const_line,
        'prompt Creating package body ' ||
        upper('&&1') || '.' ||
        upper('&&2')                    const_line,
        ''                              const_line,
        'create or replace'             const_line
from    dual;

set linesize 4000

select  text, case when line >= 2 then line+1 else line end
ord_remark
from    dba_source
where   owner = upper('&&1')
and     name = upper('&&2')
and     type = 'PACKAGE BODY'
union all
select  '/* %TC-INFO% */', 2
from    dual
order by 2;
```



```
select  '/'                                 const_line
from    dual;

spool off
```

11 Appendix B

```
select
       u.username,
       a.hash_value,
       round(a.rows_processed / a.executions) row_per_exe,
       round(a.fetches / a.executions) fch_per_exe,
       round(a.buffer_gets / a.executions) lio_per_exe,
       round(a.disk_reads / a.executions) pio_per_exe,
       round((a.cpu_time/1000) / a.executions) cpu_per_exe,
       count(p.object_name) joins,
       case
         when a.rows_processed = 0 and count(p.object_name) = 0
         then a.buffer_gets
         when a.rows_processed = 0 and count(p.object_name) > 0
         then round(a.buffer_gets / count(p.object_name))
         else round((a.buffer_gets / a.rows_processed) /
                (count(p.object_name)+1))
       end  lio_per_rowjoin,
       case
         when a.rows_processed = 0 and count(p.object_name) = 0
         then round(a.cpu_time / 1000)
         when a.rows_processed = 0 and count(p.object_name) > 0
         then round((a.cpu_time/1000) / count(p.object_name))
         else round(((a.cpu_time/1000) / a.rows_processed) /
                (count(p.object_name)+1))
       end  cpu_per_rowjoin,
       case when a.fetches = 0 then a.rows_processed
            else round(a.rows_processed / a.fetches)
       end  row_per_fch,
       a.executions,
       a.rows_processed,
       a.buffer_gets,
       a.disk_reads,
       a.cpu_time/1000 cpu_time,
       trunc(a.elapsed_time/1000) ela_time,
       a.fetches,
       trunc(a.elapsed_time / (a.cpu_time+1),2) elacpu,
       a.sql_text
from
       v$sqlarea a,
       v$sql_plan p,
       dba_users u
where
       a.executions > 0
and    a.hash_value = p.hash_value (+)
and    a.command_type not in (47,170)
and    a.parsing_user_id = u.user_id
group by
       u.username,
       a.hash_value,
       a.sql_text,
       a.executions,
       a.rows_processed,
       a.buffer_gets,
       a.disk_reads,
       a.cpu_time,
       a.elapsed_time,
       a.fetches
order by
       cpu_time desc
```


12 Appendix C

Init.ora file as represented in our source code control system:

```
File name:              adc-root/db-edb/database/admin/edb/pfile/initedb.ora
File ID:                fid.007926
Active lines:           123
Deleted lines:          284

Description:            ddelmoli 19476 Initial load
Status:                 Installed
Defined by:             ddelmoli
Defined on:             Fri Apr  4 15:51:56 2003
Installed by:           ddelmoli
Installed on:           Fri Apr  4 15:51:57 2003

           ------------ Revision History ---------------

Cset      Status Author           # Ins # Del  Dated         Description

<Base>       A   ddelmoli           187     0  Apr  4 2003   ddelmoli 19476 Initial load
aset.008442  A   ddelmoli             0     0  Apr  4 2003   Add Cset for file: database/admin/edb/pfile/initedb.ora id: fid.007926
cset.008324  A   ddelmoli            56   173  Apr 30 2003   ddelmoli 19476 aq_tm_processes and db_files
cset.008434  A   ddelmoli             5     4  May 16 2003   ddelmoli 21566 Product Configurator
cset.008687  A   ddelmoli             3     1  Aug  5 2003   ddelmoli 23378 Set audit_trail = none for Oracle bug 2685084
cset.009992  A   ddelmoli            14    23  Oct 21 2003   Update init.ora for 9i
cset.010001  A   ddelmoli             1     1  Oct 21 2003   ddelmoli 23628
cset.010750  A   ddelmoli             1     1  Oct 29 2003   ddelmoli 23628 set retention to 21600
cset.011735  A   ddelmoli            13     4  Dec  5 2003   ddelmoli 24798 Set _b_tree_bitmap_plans to false for PP queries
cset.011745  A   ddelmoli            16     7  Dec 22 2003   ddelmoli 24923
cset.011801  -   ddelmoli            10     2  Feb  2 2004   ddelmoli 25018 Requested changes from OPS
cset.011827  A   ddelmoli            10     2  Feb  5 2004   ddelmoli 25018 init.ora Changes requested by OPS
cset.012118  A   ddelmoli             5     0  Feb  8 2004   ddelmoli 24729
```

```
cset.012211  A   ddelmoli            16     2  Mar 11 2004   ddelmoli 25776 Final init.ora for MTS on EDB
cset.012299  A   ddelmoli             4     1  May 18 2004   ddelmoli 26064 Add resource_limit = true to enforce sessions per user
resource limit
cset.012456  A   ddelmoli             2     0  Jun 16 2004   ddelmoli 26925
cset.012461  A   ddelmoli             1     1  Jun 16 2004   ddelmoli 26925
cset.012481  A   ddelmoli             3     2  Jul  2 2004   ddelmoli 27106
cset.012698  A   ddelmoli             3     0  Nov 18 2004   ddelmoli 28323 Add MTS dispatchers
cset.013922  A   sbadalov            2     1  Nov  8 2005   sbadalov. TR 34281. Increase UNDO_RETENTION to 21600
cset.013970  A   ddelmoli            55    51  Jan 11 2006   ddelmoli 34646

 Line   ION  Cset                ----- active Source -----

    1 00026  <Base>             #
    2 00188  cset.008324        # Init.ora file for EDB
    3 00338  cset.012299        # %TC-INFO%
    4 00029  <Base>             #
    5 00353  cset.013970        # Change Log moved to end of file
    6 00354  cset.013970
    7 00033  <Base>             #
    8 00355  cset.013970        # MTS configuration
    9 00356  cset.013970        #
   10 00357  cset.013970        # IP addresses are the addresses of the adapters used for this instance
   11 00358  cset.013970        # on this server and must be localized
   12 00359  cset.013970        #
   13 00360  cset.013970
   14 00361  cset.013970        dispatchers = '(ADDRESS=(PROTOCOL=TCP)...'
   15 00362  cset.013970        dispatchers = '(ADDRESS=(PROTOCOL=TCP)...'
   16 00363  cset.013970        dispatchers = '(ADDRESS=(PROTOCOL=TCP)...'
   17 00364  cset.013970
   18 00365  cset.013970        shared_servers                   = 5
   19 00366  cset.013970        max_shared_servers               = 30
   20 00310  cset.011827
   21 00311  cset.011827        # BUG 3372713 01/28/04
   22 00312  cset.011827        event                           = "10262 trace name context forever, level 4000"
   23 00313  cset.011827
```

```
24 00191  cset.008324        db_name                          = edb
25 00256  cset.009992        db_domain                        = pdedb.prod.netsol.com
26 00193  cset.008324        global_names                     = true
27 00194  cset.008324        control_files                    = (/edb/oralog01/edb/cntrledb.dbf,
28 00195  cset.008324                                            /edb/oralog02/edb/cntrledb.dbf)
29 00134  <Base>
30 00257  cset.009992        undo_management                  = auto
31 00258  cset.009992        undo_tablespace                  = EDB_UNDO01
32 00351  cset.013922        #undo_retention                   = 3600
33 00352  cset.013922        undo_retention                   = 21600
34 00136  <Base>
35 00261  cset.009992        compatible                       = 9.2.0
36 00203  cset.008324        optimizer_mode                   = choose
37 00204  cset.008324        optimizer_index_cost_adj         = 40
38 00205  cset.008324        optimizer_index_caching          = 90
39 00206  cset.008324        query_rewrite_enabled            = true
40 00280  cset.011735        _b_tree_bitmap_plans             = false
41 00142  <Base>
42 00207  cset.008324        db_block_size                    = 8192
43 00314  cset.011827        db_cache_advice                  = ready
44 00367  cset.013970        sga_max_size                     = 2G
45 00368  cset.013970        db_cache_size                    = 640M
46 00369  cset.013970        shared_pool_size                 = 400M
47 00370  cset.013970        shared_pool_reserved_size        = 32M
48 00371  cset.013970        java_pool_size                   = 32M
49 00372  cset.013970        large_pool_size                  = 128M        # 256 MTS sessions @ 500K per
50 00146  <Base>
51 00264  cset.009992        workarea_size_policy             = auto
52 00315  cset.011827        pga_aggregate_target             = 975M
53 00149  <Base>
54 00212  cset.008324        processes                        = 4000
55 00296  cset.011745        transactions                     = 6000
56 00214  cset.008324        session_cached_cursors           = 100
57 00215  cset.008324        open_cursors                     = 1000
58 00217  cset.008324        max_enabled_roles                = 140
```

```
59 00341  cset.012299        resource_limit                   = true
60 00152  <Base>
61 00219  cset.008324        parallel_max_servers             = 40
62 00156  <Base>
63 00222  cset.008324        log_checkpoint_interval          = 10000
64 00223  cset.008324        log_checkpoint_timeout           = 0
65 00225  cset.008324        log_archive_start                = true
66 00226  cset.008324        log_archive_dest                 = /edb/oraarch/edb
67 00227  cset.008324        log_archive_format               = "edb%s.ARC"
68 00344  cset.012461        utl_file_dir                     = /app/oracle/admin/edb/utl_file
69 00162  <Base>
70 00229  cset.008324        job_queue_processes              = 3
71 00231  cset.008324        aq_tm_processes                  = 2
72 00163  <Base>
73 00232  cset.008324        db_files                         = 1000
74 00233  cset.008324        db_file_multiblock_read_count    = 32
75 00235  cset.008324        fast_start_parallel_rollback     = high
76 00237  cset.008324        db_writer_processes              = 4
77 00166  <Base>
78 00238  cset.008324        timed_statistics                 = true
79 00250  cset.008687        # Bug 2685084 ORA-00600 [15056],ORA-00600 [13262] WHEN AUDIT_TRAIL=TRUE 08/05/03
80 00251  cset.008687        audit_trail                      = none
81 00169  <Base>
82 00316  cset.011827        _trace_files_public              = true
83 00240  cset.008324        background_dump_dest             = /app/oracle/admin/edb/bdump
84 00241  cset.008324        core_dump_dest                   = /app/oracle/admin/edb/cdump
85 00242  cset.008324        user_dump_dest                   = /app/oracle/admin/edb/udump
86 00243  cset.008324        max_dump_file_size               = 10485760
87 00317  cset.012118
88 00373  cset.013970        # 04/30/03  Initial creation / copy from Production (ddelmolino / jchen)
89 00374  cset.013970        # 05/16/03  Remove rollback segments from init.ora (ddelmolino)
90 00375  cset.013970        # 08/05/03  Change audit_trail to none (from DB) (ddelmolino)
91 00376  cset.013970        # 10/21/03  Updated for 9i (ddelmolino)
92 00377  cset.013970        # 12/05/03  Added _b_tree_bitmap_plans = false (ddelmolino)
93 00378  cset.013970        #           The default for this setting was false in 8i and became
```

```
 94 00379  cset.013970        #           true in 9i.  It is affecting queries against tables which
 95 00380  cset.013970        #           have many single-column indexes and where those queries
 96 00381  cset.013970        #           have equality or range predicates on more than one index.
 97 00382  cset.013970        #           In particular, queries against WPI.  It appears to be making
 98 00383  cset.013970        #           the wrong decision about converting b-trees to bitmaps
 99 00384  cset.013970        #           looks like its using the wrong cardinality estimates.
100 00385  cset.013970        # 12/19/03  Remove undo_retention from init.ora (jchen)
101 00386  cset.013970        # 12/19/03  Remove resource_manager_plan from init.ora (jchen)
102 00387  cset.013970        # 12/19/03  Remove transactions_per_rollback_segment from init.ora (jchen)
103 00388  cset.013970        # 12/19/03  Add db_cache_advice to init.ora (jchen)
104 00389  cset.013970        # 12/19/03  Add shared_pool_reserved_size to init.ora (jchen)
105 00390  cset.013970        # 12/19/03  Add transactions to init.ora (jchen)
106 00391  cset.013970        # 12/19/03  Increase shared_pool_size from 320m to 350m (jchen)
107 00392  cset.013970        # 12/19/03  Increase db_cache_size from 256m to 350m (jchen)
108 00393  cset.013970        # 12/19/03  Reduce pga_aggregate_target from 2560m to 1300m (jchen)
109 00394  cset.013970        # 01/28/04  Reduce pga_aggregate_target to 975M from 1300M (ddelmolino)
110 00395  cset.013970        #           Added _trace_files_public = true (ddelmolino)
111 00396  cset.013970        #           Added event 10262 for bug 3372713
112 00397  cset.013970        # 03/11/04  Added in dispatchers and shared_servers to support MTS
113 00398  cset.013970        #           Set large_pool_size to 64M from 0 for MTS (ddelmolino)
114 00399  cset.013970        # 03/20/04  Increased undo_retention to 3600 (1 hour) (jchen)
115 00400  cset.013970        # 05/18/04  Added resource_limit = true to enforce sessions limit
116 00401  cset.013970        #           (ddelmolino)
117 00402  cset.013970        # 07/02/04  Increase large_pool_size to 128M from 64M (ddelmolino)
118 00403  cset.013970        # 11/18/04  Add dispatcher lines to increase from 1 to 3 (ddelmolino)
119 00404  cset.013970        # 11/21/05  Add sga_max_size at 2GB to enable dynamic cache allocation (ddelmolino)
120 00405  cset.013970        #           Set default db_cache_size at 640M (from 350M)
121 00406  cset.013970        #           Set shared_pool_size at 400M from (350M)
122 00407  cset.013970        #           Set java_pool_size at 32M (default)
123 00336  cset.012211        #
```